



Desarrollo y gestión ágil de proyectos Python

Lección 4: Medición y estimación ágil

ÍNDICE

Medición y estimación ágil	3
Presentación y objetivos.....	3
1. Un poco de historia.....	5
1.1. Contextualización	5
2. Medición ágil.....	12
3. Unidades relativas	15
3.1. Puntos de historia.....	15
3.2. Tiempo real y tiempo ideal.....	16
3.3. Velocidad.....	17
4. Estimación ágil.....	19
4.1. Técnicas ágiles de estimación.....	19
4.1.1. Planning Poker.....	20
4.1.2. Tallas de camiseta	22
4.1.3. Puntos de votación (dot voting)	23
4.1.4. El sistema del cubo	23
4.1.5. Grande / Incierto / Pequeño.....	25
4.1.6. Mapeo de afinidad	26
4.1.7. Método de ordenamiento.....	27
5. Operativa de estimacion con planning poker	28
6. Puntos clave.....	31

Medición y estimación ágil

«Medir el progreso de programación por líneas de código es como medir el progreso de la construcción de aviones por el peso» ~ Bill Gates

PRESENTACIÓN Y OBJETIVOS

Uno de los primeros conceptos desarrollados por el ser humano fue el de número, pues tenía la necesidad de poder expresar numéricamente todo lo que se encontraba a su alrededor.

Entonces comenzó a medir mediante un simple conteo de objetos. Más tarde, y por propias necesidades de su desarrollo, enunció el concepto de medida, realizando las primeras mediciones a partir de unidades muy rudimentarias.

La medición es un concepto complejo, en el sentido de que es necesario encontrar un patrón objetivo.

En el caso del software, algo ya de por sí abstracto para algunas personas, es más complicado aún si cabe.

No obstante, desde los orígenes del software se ha prestado atención a este área de la ingeniería del software.

En esta lección vamos a explicar los conceptos básicos sobre medición y estimación ágil que conviene conocer.



Objetivos

En esta lección aprenderás:

- Cómo medir de manera ágil.
- Las técnicas de estimación ágiles existentes.
- La operativa habitual de estimación en Scrum.

1. UN POCO DE HISTORIA

1.1. Contextualización

El ambiente caótico que caracterizó las primeras etapas de las técnicas de codificación desembocó en la inconformidad creciente de los usuarios, así como en proyectos excedidos en tiempos de entrega y presupuestos, casi de forma masiva, causas que marcaron la pauta en la “búsqueda de alternativas para esquematizar de alguna manera la producción del Software”.

Mediados de los 60 toma popularidad el término “crisis del Software” para referirse a los problemas recurrentes que caracterizaba los procesos de desarrollo, en costos, fiabilidad, entregas extemporáneas e insatisfacción de los clientes; aspectos que se acuñaron como “síntomas de la crisis del Software.”

Entre otras dificultades prácticas que acusaban esta la alta calificación que demandaban los sistemas para con los usuarios y lo costoso y complejo que resultaba su mantenimiento.

El término ingeniería del software empezó a usarse con más énfasis a finales de la década de los 70, para representar ya para ese momento un área de conocimiento que se estaba desarrollando en torno a las problemáticas que ofrecía el software y tomando base en los postulados de Dijkstra.

Dijkstra junto a otros autores publicaría luego el artículo “Go to statement considered harmful” y junto a su libro “Discipline of Programming” estableció ciertos parámetros para el desarrollo del Software de forma exitosa que actualmente siguen siendo legado, algunos de sus postulados resumidos son:

- | El coste del desarrollo inicial debe ser relativamente bajo.
- | El software debe ser fácil de mantener.
- | Cualquier desarrollo debe de ser portable a nuevo hardware.
- | El software debe hacer lo que el cliente quiere.

- | El desarrollo de programas mediante top-downy en contraposición a bottom-up. El desarrollo debe seguir un conjunto de pasos formales para descomponer los problemas grandes (lema divide y vencerás).

Dicho esto, es decir, una vez comprendidos los antecedentes históricos que no ha llevado hasta aquí, hay que tener también en consideración que a las organizaciones de desarrollo software y, obviamente, a sus clientes les preocupa la calidad, tiempo de entrega, coste, el rendimiento del software y la productividad de sus equipos de desarrollo.

Esta preocupación no es algo novedoso que haya surgido a raíz de la aplicación de marcos de trabajo ágiles; sino que viene de antaño. Por lo que, a lo largo de la historia del software son varias y muy diversas las medidas que se vienen aplicando, algunas con mayor exactitud, fiabilidad y trascendencia que otras.

- | Una medida proporciona una indicación cuantitativa de extensión, cantidad, dimensiones, capacidad y tamaño de algunos atributos de un proceso o producto.,,,
- | Métrica es una medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo dado (IEEE, 1993),,,
- | Mientras que el indicador es una métrica o combinación de métricas que proporcionan una visión profunda, del proceso de software, del proyecto de software o del producto en sí (Ragland, 1995).

Medición y Métricas

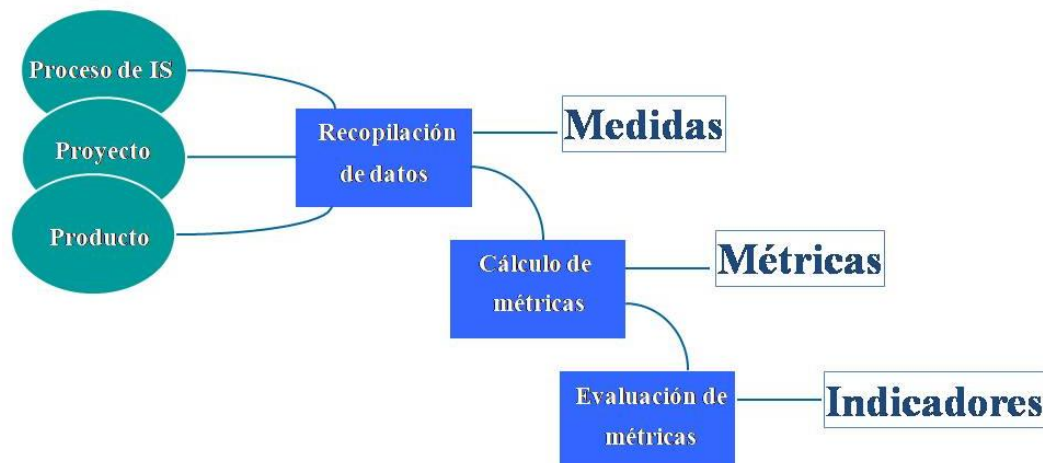


Ilustración 1.1. Métricas software

Existen medidas tanto directas como indirectas.

Medidas del software

Medidas Directas

- Coste.
- Esfuerzo humano.
- Líneas de código.
- Velocidad de ejecución.
- Tamaño de memoria.
- Número de defectos.
- Etc.

Medidas Indirectas

- Funcionalidad.
- Calidad.
- Complejidad.
- Eficiencia.
- Fiabilidad.
- Facilidad de uso.
- Etc.

Ilustración 1.2. Medidas del software

Una de las medidas directas que se aplicaba antiguamente es la número de líneas de código.

$$\text{Productividad} = \text{KLDC} / \text{PM}$$

Donde:

LDC = líneas de código

KLDC= miles de líneas de código

PM=persona-mes (originalmente, incluso se hablaba de MM, men-month)

A principios de 1982, en Apple tenían mucha prisa por sacar el Lisa, así que decidieron “motivar” a sus programadores y se les ocurrió medir el progreso de éstos basándose en el número de líneas de código producidas semanalmente.

Esta historia real viene a confirmar algo que hoy día ha quedado más que demostrado, el trabajo de un programador no es cuantificable como el número de barras de pan que haga un panadero o el de ladrillos que ponga un albañil. El trabajo de un programador debe medirse en eficacia y eficiencia, y medir líneas de código resulta bastante absurdo.

Otra medida muy utilizada y que está quedando desfasada con el auge del desarrollo ágil de proyectos es la HM (**Hora-Hombre**).

Para calcular el esfuerzo que un programador realiza, se utilizaba una medida muy conocida denominada hora-hombre. Su cálculo es muy sencillo, y mide la cantidad de tiempo que emplea un programador realizando su trabajo de desarrollo software, lo cual permite determinar el número de programadores (o el número de horas) necesarios para completar el proyecto en un tiempo concreto.

Así, la hora-hombre se utiliza con el fin de ver la aportación de un programador medio en la consecución de un objetivo de desarrollo planteado. Evidentemente las horas-hombre hacen referencia al trabajo ininterrumpido de los empleados, ya que no incluyen períodos de interrupción de la actividad como pausas para comer, ir al baño o tiempos de descanso.

El cálculo de las horas hombre es bastante sencillo. Para ello, necesitas saber cuál es la cantidad de horas de trabajo efectivo que realiza un programador cada día, sin contar su tiempo libre ni posibles vacaciones o descansos. En una compañía cuyos empleados tienen una jornada de 40 horas semanales y trabajan de lunes a viernes, podemos pensar que cada uno de ellos emplea 8 horas al día en trabajar.

Sin embargo, esto nunca termina siendo del todo real, ya que entre los descansos para comer, alguna pequeña distracción y el surgimiento de imprevistos, la jornada laboral nunca se corresponde exactamente a las horas realmente trabajadas. Si estimamos que el trabajador pierde unos 45 minutos cada día en este tipo de cosas, trabajaría de forma efectiva unas 7,25 horas hombre diarias. Si lo multiplicamos por 5 días, cada empleado realizaría 36,25 horas hombre semanales. Imaginando que en esta compañía trabajan 30 empleados, su rendimiento semanal en cuanto a horas hombres sería de 1087,5.

En 1975, cuando la denominada “guerra de los métodos de Ingeniería de Software” estaba en la cima (denominada así, dada la enorme cantidad de métodos propuestos con notaciones y procesos diferentes, tratando de imponerse unos sobre otros), Fred Brooks publicó su famoso libro *The Mythical Man-Month* (El Mítico Hombre-Mes, en alusión directa a la unidad de medida de esfuerzo de Horas-Hombre, Hombre-Mes, etc.).

Brooks aprendía su lección acerca del desarrollo de software cuando él era el administrador del probablemente primer proyecto emprendido de desarrollo de software de gran escala, el desarrollo del sistema operativo IBM/360 (OS/360) a inicio de los años 60. El centro del mensaje de Brooks es que el desarrollo de software es un proceso centrado en el ser humano. Atendiendo a los innumerables problemas y fracasos del desarrollo de software sumado a una baja productividad de los programadores y una mala calidad de los productos de software, Brooks acuñó la frase hasta hoy usada: “La crisis del software”, aunque algunos plantean que se debería hablar de una “aflicción crónica”, pues las crisis tienen un punto alto y luego se resuelven, a diferencia de lo que sucede con el desarrollo de software.

Su libro dispuso la primera metodología de desarrollo de software útil, porque el método presionaba por administrar procesos de personas y no procesos de ingeniería mecanizados.

Brooks, en un estilo humorístico, identifica los muchos problemas que presentan los proyectos de software.

Cualquiera que haya estado involucrado en un proyecto de desarrollo de software los encontrará relevantes incluso hoy día.

1. La mina de alquitrán: “Los proyectos de software son quizás lo más intrincado y complejo de las cosas que hace la humanidad” (en términos de las distintas clases de partes que lo componen). Esto es más cierto hoy día en varias órdenes de magnitud, dado el aumento considerable de complejidad de las aplicaciones.
2. El Mítico Hombre-Mes: “Muchos proyectos fracasan más por pérdidas de tiempo calendario, que por todas las otras razones combinadas”. Esta sentencia, que abre el libro de Brooks, es cierta hasta ahora. No se ha desarrollado hasta ahora una buena forma de estimar cuánto tiempo tomarán los proyectos de programación. La programación es un proceso creativo similar al arte y la música. Los proyectos de programación son tentativas comerciales. Esto lleva al corazón del problema: ¿Cómo administramos exitosamente un proceso humano creativo (como opuesto a procesos mecánicos y productivos)?
3. La Ley de Brooks (la que exige ser validada por investigaciones): “Agregar horas-hombre a un proyecto atrasado hace que se retrase más”. Las personas y los tiempos no son intercambiables. Hay cierto procesos que no pueden ser apurados. Agregar más personas incrementa las intercomunicación y la sobrecarga en entrenamiento, tanto como interrumpir el avance.
4. El Efecto del segundo sistema: “El Segundo es el sistema más peligroso de una persona que sobre diseña; la tendencia generales a sobre diseñarlo”. En lenguaje moderno debería llamarse “featuritis”.

5. ¿Por qué la Torre de Babel falla?: “Desastres de calendario, inadaptación funcional y errores de sistema, todos crecen por-que la mano izquierda no sabe que está haciendo la mano derecha. Los equipos definen supuestos”. La comunicación es la parte más difícil de cualquier emprendimiento humano. Esto, por supuesto, es el centro del Mítico Hombre-Mes.
6. Un paso adelante y un paso atrás: “La entropía del sistema crece en el tiempo”. Esto también es conocido como “Pedazos de tonterías”. Reparar tiende a destruir estructuras e incrementar el desorden.

El Impacto del Mítico Hombre-Mes es innegable. Junto al crecimiento de los problemas en el desarrollo de software, el libro ofrece algunas soluciones importantes que, cuando son tomadas juntas, proporcionan un método universal para el desarrollo de software. Sin embargo, The Mythical Man-Month, a diferencia de la programación estructurada, no formó las bases de ninguna metodología promovida en los años 70 u 80. Mientras el libro fue de una importancia increíble, no fue hasta finales de los años 90 que las ideas de Brooks fueron tomadas por los métodos ágiles.

2. MEDICIÓN ÁGIL

El objetivo de scrum es producir el mayor valor posible de forma continua, así que cabe preguntarse siempre cómo contribuye el uso del indicador en el valor que se proporciona al cliente.

Medir es costoso y debe servir a un propósito mayor, no convertirse en un fin en sí mismo.

Los objetivos que perseguimos con las herramientas de estimación que vamos a ver son:

- | Poder planificar la duración de cada sprint de forma realista.
- | Marcar el ritmo de avance (sobre todo en equipos que están empezado a trabajar de forma ágil).
- | Sincronizar equipos.
- | Cerrar fechas de entrega.

En fundamental tener en cuenta estos dos conceptos clave:

1. No se mide el trabajo realizado, sino el que queda.
2. Se mide empleando unidades relativas.



Presta atención

1. No se mide el trabajo realizado, sino el que queda.
2. Se mide empleando unidades relativas.



Medir el trabajo puede ser necesario por dos razones:

1. Para registrar el que ya se ha hecho
2. Para estimar por adelantado el que se debe realizar.

En ambos casos se necesitan una unidad y un criterio objetivos de cuantificación.

Medir el trabajo ya realizado no entraña dificultad. Se puede hacer con unidades relativas al producto, como historias completadas; o a los recursos, como coste o tiempo de trabajo.

Pero la gestión de proyectos ágil no mide el trabajo ya hecho para calcular el avance del trabajo; es decir, restándolo del tiempo previsto.

Por ejemplo: "Esto debía costar una semana. Como han pasado tres días, quedan cuatro para que esté terminado."

Esto, que suena lógico, en la realidad no se suele cumplir. Surgen imprevistos o se encuentran atajos que hacen que el tiempo estimado al principio no sea exacto. Teniendo esto en cuenta, no se determina el grado de avance por el trabajo realizado, sino por el que queda pendiente.

Asno lo tenía claro cuando viajaba hacía el reino Muy-muy-lejano, ¿qué es lo que pregunta, una y otra vez? ¿Cuánto tiempo lleva viajando? No ... ¡eso no le importa! La pregunta es: "¿falta mucho?", es decir, ¿cuánto queda?



Ilustración 2.1 Escena "Falta mucho" Shrek 2

Es posible que otros procesos de la organización necesiten registrar el esfuerzo invertido, pero calcular el avance del proyecto es diferente.



No te olvides...

Scrum mide el trabajo pendiente, para:

- 1. Estimar el esfuerzo y tiempo previstos para realizar determinadas tareas, historias de usuario y epics (historias de gran tamaño).**
- 2. Determinar el grado de avance del proyecto, y en especial de cada sprint.**

3. UNIDADES RELATIVAS

3.1. Puntos de historia

El trabajo necesario para realizar un requisito o una historia de usuario no se puede prever de forma absoluta porque rara vez son realidades de una solución única.

En el caso de que se pudiera, por otra parte, la complejidad de la medición haría una métrica demasiado pesada para la gestión ágil.

No resulta posible estimar con precisión la cantidad de trabajo que hay en un requisito. En consecuencia, tampoco se puede saber con antelación cuánto tiempo exigirá, porque a la incertidumbre del trabajo se suman las inherentes al tiempo, es decir, no se puede estimar la cantidad o la calidad del trabajo que realiza una “persona-media por unidad de tiempo”, porque son muy grandes las diferencias de unas personas a otras. Es más, la misma tarea realizada por la misma persona requerirá diferentes tiempos dependiendo de las circunstancias.

Por todas estas razones, al estimar de forma ágil, se prefiere emplear unidades relativas. En gestión ágil se suelen emplear puntos como unidad de trabajo, usando denominaciones como puntos de historia.



No te olvides...

En gestión ágil se suelen emplear puntos de historia como unidad de trabajo.

Cada organización, según sus circunstancias y su criterio, institucionaliza su métrica de trabajo, su punto. Es el tamaño relativo de tareas que se suele emplear. Es importante que el significado y la forma de aplicar la métrica sea siempre la misma en las mediciones de la organización, y que sea conocida por todos.

El tipo de punto dependerá de la organización. En un equipo de programación el punto puede ser equivalente a preparar una pantalla de login; para un equipo de diseño gráfico, la maquetación de un tríptico.

El punto ayuda, por un lado, a dimensionar la estimación de una tarea comparándola con una ya conocida, y por otro lado, a contrastar la dificultad que la tarea presenta para cada miembro del equipo según sus especialidades. Un ejemplo para ilustrar esto último podría ser el esfuerzo que cuesta freír un huevo.

Si se estima cuántos huevos fritos costaría planchar una camisa, la respuesta dependerá de la persona. Alguien puede ser muy habilidoso friendo huevos, pero muy torpe para planchar camisas, y estimará que eso le costaría 8 huevos fritos; es decir, 8 puntos. Alguien muy acostumbrado a las tareas domésticas, en cambio, podría estimar la tarea en un punto o un huevo frito.

Ambos tienen razón: la cuestión es que la persona que estime sea la que va a realizar la tarea.

Los puntos de historia suelen ser una unidad relativa o abstracta basada en algo con lo que el equipo esté muy familiarizado.

3.2. Tiempo real y tiempo ideal

Cuando se calcula el calendario de un sprint tendemos a estimar el esfuerzo en tiempo ideal, que es el tiempo de trabajo en condiciones ideales. Es decir, es lo que nos costaría realizar una tarea en un estado de flujo, concentrados y sin ninguna distracción o impedimento.

Es importante ser conscientes de la diferencia por tanto entre el tiempo ideal y el tiempo real a la hora de estimar.

Un buen ejemplo de esto, es un partido de baloncesto. Si nos preguntan cuánto dura un partido de baloncesto, la respuesta en tiempo exacto de juego es 40 minutos, pero el tiempo real de un partido de baloncesto suele ser de más de una hora, pues no puede terminar en empate. Se alarga por los tiempos muertos, las faltas, el entretiempo del partido y las prórrogas.



Ilustración 3.1. Marcador baloncesto

Todos sabemos que es normal que a veces un desarrollo, que podría hacerse en una hora de tiempo ideal, acabe ocupando media jornada de trabajo. Manejar estos dos conceptos de tiempo puede ayudarnos a organizar el trabajo de manera más objetiva y evitar el estrés de metas inalcanzables.

3.3. Velocidad

Los puntos de historia permitirán medir la velocidad que tiene el equipo completando objetivos a lo largo de las iteraciones (puntos de historia por iteración) y de esta manera ir proyectando el final del proyecto.

La velocidad es igual a la cantidad de trabajo realizado por el equipo en un sprint.

Así, por ejemplo, una velocidad de 150 puntos indica que el equipo realiza 150 puntos de trabajo en cada sprint.

No obstante, al salir del marco estándar de scrum podemos encontrar sprints de diferentes duraciones. Cuando esto sucede, se puede expresar la velocidad en unidades de tiempo en lugar de por sprint. Es decir: “la velocidad media del equipo es de x puntos por semana”.

Las 2-3 primeras iteraciones la velocidad es más inestable (debido a la complejidad propia de un proyecto: dominio/ requisitos nuevos, tecnología nueva, personas nuevas) y después se va estabilizando, con lo que ya es posible utilizarla para hacer predicciones. Si no se estabiliza, posiblemente es porque existen problemas subyacentes que lo impiden (en la organización, en la estabilidad del equipo, etc.) y que habrá que solucionar.

Hay que tener en consideración que las estimaciones (y la velocidad) se ven afectadas cuando se introducen nuevas tecnologías o conceptos de negocio en medio del proyecto, aunque después se vuelven a estabilizar.

Esto es debido a que en estos casos se introduce incertidumbre y esta incertidumbre debe ser considerada en las estimaciones.

$$\text{Velocidad del equipo} = \text{Story points} / \text{iteración}$$

4. ESTIMACIÓN ÁGIL

La principal premisa en que se basa la estimación ágil es el conocimiento, la experiencia del propio equipo.

Para lograr buenas estimaciones se requiere de experiencia del Scrum Team.

Hay un parámetro base en Scrum: cómo medir la cantidad de trabajo que hay que hacer.

Este parámetro es fundamental a la hora de planificar y acometer el trabajo.

Para ello, es necesario establecer una unidad de medida relativa, como ya hemos comentado anteriormente.

Scrum recomienda utilizar los puntos historia (story points).

**Story Point = Esfuerzo necesario +
complejidad + riesgo ...**

4.1. Técnicas ágiles de estimación

Como hemos visto en la lección anterior, la estimación es parte de la información esencial en una historia de usuario.

Estimar el esfuerzo ayuda al propietario de producto a priorizar y al equipo a decidir qué historias de la pila caben en el sprint, es decir, a qué trabajo se comprometen y la dirección a planificar el trabajo global.

Un hecho a considerar es que la incertidumbre para estimar historias de usuario pequeñas es mucho menor que para historias grandes.

Además, aunque el tamaño de las historias crece linealmente, la incertidumbre lo hace exponencialmente.

Existen varias técnicas que permiten realizar la estimación de tiempo, aunque las más usadas son planning poker con Fibonacci para las historias de usuario y tallas de camiseta para epics.

4.1.1. Planning Poker

La técnica de planning poker permite hacer una estimación inicial del proyecto rápida y fiable, dado que todos los miembros del equipo comparten sus diferentes informaciones y expresan su opinión sin sentirse condicionados por el resto.

Es el método de estimación **más usado habitualmente**.

Es un método basado en la serie de Fibonacci, básicamente es un juego creado por James Grening que consiste en el levantamiento de cartas por cada participante.

Como hemos comentado anteriormente, el tamaño de las historias crece linealmente, la incertidumbre lo hace exponencialmente.

Una serie numérica que se presta bien a esta tarea es la de Fibonacci, compuesta por números que son la suma de los dos anteriores. La distancia entre los números de la serie refleja el hecho de que la incertidumbre inherente a la estimación crece con el tamaño de la historia. Así, las diferencias entre 1, 2 y 3 puntos de historia son probablemente mejor comprendidas que las diferencias entre 13, 21 y 42.

1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, ...

También hay razones que derivan de la naturaleza humana para utilizar esta serie. Nos es fácil estimar determinando cuánto menor o mayor es algo respecto a otra cosa, por lo que en scrum y otros métodos ágiles se emplean técnicas de estimación relativa. Pero tenemos la tendencia de pensar en múltiplos de dos.

Por ejemplo, comparamos una historia de usuario con una de 2 puntos, vemos que es mayor y automáticamente pensamos en 4, 8 o 16. La serie de Fibonacci nos obliga a romper con este hábito y buscar la estimación más adecuada.

Las barajas que suelen usarse en estimaciones basadas en la serie de Fibonacci pueden contener variaciones. Algunas incluyen el 0 para historias que requieren un esfuerzo casi nulo y 1/2 para historias muy pequeñas. Algunas barajas sustituyen el 21 por un 20, ya que decir que una historia tiene un tamaño de 21 da una falsa sensación de precisión. A partir del 21 las barajas suelen pasar a incluir un símbolo de infinito (∞) y, si siguen, incluyen valores redondos como 40 y 100.



Ilustración 4.1.. Cartas de estimación Fibonacci Scrum

El objetivo del planning poker no es hacer predicciones que puedan confirmarse más adelante, sino ofrecer un proceso de estimación rápido, que dé al equipo valores útiles para estimar la pila de producto o planificar el sprint. Más importante aún es lograr que todos comprendan el elemento estimado: si los participantes estiman con valores muy dispares es que alguien sabe algo que otros no. Recordemos que en los equipos hay especialistas de diferentes áreas, optimistas y pesimistas, y eso hace que algunos vean soluciones que otros no ven. Estas discusiones servirán para alinear el conocimiento de todo el equipo.

4.1.2. Tallas de camiseta

Esta es una técnica muy buena para estimar un backlog grande de producto. Especialmente cuando tenemos varios equipos concurrentes trabajando en el mismo producto.

Es una manera informal y rápida de tener una idea aproximada del esfuerzo requerido para hacer algo, lo que la hace ideal para la estimación de elementos grandes como los epics y temas, para los que la serie de Fibonacci no es adecuada, ya que sus números darían una idea de precisión inexistente.

Esta solución de estimación consiste en emplear una técnica basada en tallas de camisa.

Las tallas sirven para intuir más que estimar, con un alto grado de incertidumbre, tal y como requieren elementos demasiado grandes y poco definidos.

Esta técnica tiene la ventaja de ser puramente relativa, al no poder traducir sus valores a tiempo, jornadas u horas.

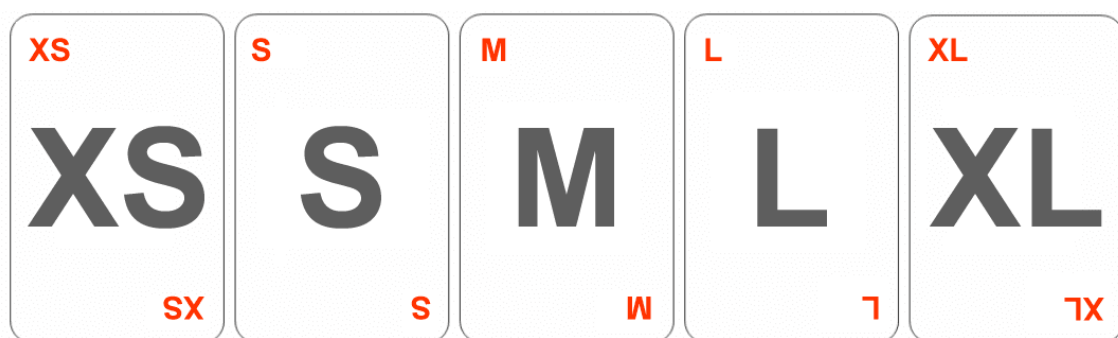


Ilustración 4.2.. Cartas de estimación tallas de camiseta Scrum

4.1.3. Puntos de votación (dot voting)

La votación por puntos es una forma de votación acumulativa.

Es un método para identificar problemas o priorizar un gran número de opciones o ideas en un equipo.

Los participantes de la votación por puntos pueden distribuir un número predeterminado de puntos o puntos a las opciones disponibles. Todos los votos de los miembros del equipo se evalúan por igual y se encuentra rápidamente un consenso.

Por lo general, la votación por puntos se lleva a cabo de la siguiente manera:

- Recoge opciones, ideas o problemas.
- Todos los participantes reciben un número fijo de puntos.
- Cada participante distribuye sus puntos a las opciones ofrecidas. Los participantes pueden dar una opción cualquiera o un número predeterminado de sus puntos.
- La opción con más puntos gana.

Este método suele ser útil cuando nos enfrentamos a un conjunto relativamente pequeño de elementos y necesitamos una técnica súper simple y efectiva para estimar.

El método se originó a partir de la toma de decisiones. Funciona bien tanto para equipos pequeños como para grandes, pero tenemos que limitar el número de elementos estimados.

4.1.4. El sistema del cubo

Este sistema es una alternativa al estimar un gran número de elementos con un gran grupo de participantes.

El sistema Bucket de estimación funciona de la siguiente manera:

1. Configurar el entorno físico según el diagrama que se muestra a continuación. Asegurándose de que todos los elementos a estimar estén escritos en tarjetas.

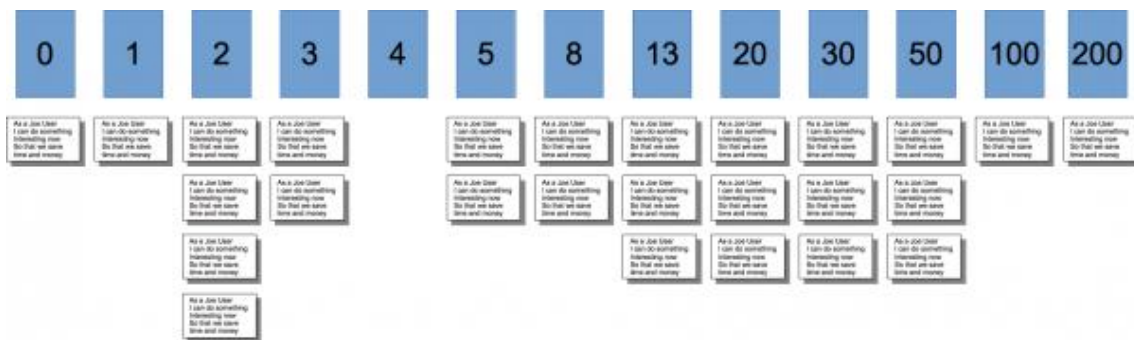



Ilustración 4.3.. Sistema del cubo

2. Elegir un artículo al azar de la colección. Leerlo al grupo. Colocarlo en el cubo de "8". Este artículo es nuestro primer artículo de referencia.
3. Elegir otro artículo al azar de la colección. Leerlo al grupo. El grupo discute su posición relativa en la escala. Una vez que se haya alcanzado el consenso, colocar el artículo en el cubo correspondiente.
4. Elegir un tercer elemento al azar y, una vez que se haya alcanzado el debate y el consenso, colocar en el cubo correspondiente.
5. Si los elementos aleatorios han sesgado claramente la escala hacia un extremo o hacia el otro, volver a escalar los elementos (por ejemplo, si el primer elemento es realmente muy pequeño y debería estar en el cubo "1").
6. "Divide y vencerás". Asignar todos los elementos restantes por igual a todos los participantes. Cada participante coloca elementos en la escala sin discutirlos con otros participantes. Si una persona tiene un artículo que realmente no comprende, entonces ese artículo se puede ofrecer a otra persona.

7. ¡Prueba de cordura! Todos revisan en silencio los elementos de la escala. Si un participante encuentra un elemento que cree que está fuera de lugar, puede comunicárselo al grupo. Luego, el grupo lo discute hasta que se llega a un consenso y se coloca en uno de los cubos.
8. Finalmente, hay que escribir los números de los cubos en las tarjetas para que se registren las estimaciones.



Para más información

<http://www.agileadvice.com/2013/07/30/referenceinformation/agile-estimation-with-the-bucket-system/>

4.1.5. Grande / Incierto / Pequeño

Un método muy rápido de estimación aproximada es el método Grande / Incierto / Pequeño.

Se le pide al equipo que coloque los artículos en una de estas categorías.

El primer paso es categorizar los elementos obvios en las dos categorías extremas (Grande y Pequeño).

A continuación, el equipo puede discutir los elementos más complejos. Esto es en realidad una simplificación del sistema de cubos.

El sistema es especialmente bueno para usar en equipos más pequeños con elementos comparables. Como siempre, podemos asignar tamaños numéricos a estas 3 categorías.

4.1.6. Mapeo de afinidad

El mapero de afinidad es una herramienta desarrollada por Jiro Kawakita que permite categorizar información, organizando un conjunto de ideas o datos al encontrar las relaciones existentes entre ellos y hacer conjuntos visuales a partir de estas relaciones.


Ayudan a sintetizar la información que hay sobre un tema, para ello se utiliza un diagrama de afinidad.

Por lo tanto, el diagrama de afinidad es una herramienta para categorizar datos, información o ideas basados en la relación que tienen entre sí.

Este método es útil cuando:

- Se quiere organizar un conjunto amplio de datos.
- Se pretende abordar un problema de manera directa.
- El tema sobre el que se quiere trabajar es complejo.
- Es necesario el consenso del grupo.

Funciona mejor con un grupo pequeño de personas y un número relativamente pequeño de elementos.



Para más información

Betancourt, D. F. (12 de diciembre de 2016). Diagrama de Afinidad: El método KJ paso a paso con ejemplo detallado. Recuperado el 07 de junio de 2021, de Ingenio Empresa: www.ingenioempresa.com/diagrama-de-afinidad.

4.1.7. Método de ordenamiento

Éste es un ejercicio donde se obtiene una imagen precisa del tamaño relativo de los elementos.

Funciona mejor con un pequeño grupo de expertos.

Todos los elementos se ponen en orden aleatorio en una escala que va de “pequeña” a “grande”.

A cada participante se le pide que mueva un elemento en la escala.

Cada movimiento es solo un punto más bajo o uno más alto en la escala (muy pequeños, pequeños, medianos, grandes, muy grandes), o simplemente el participante cede el turno.

Esto continúa hasta que ningún miembro del equipo quiera mover elementos o ceda su turno.

5. OPERATIVA DE ESTIMACION CON PLANNING POKER

Como ya se ha comentado, la estimación de póquer es una práctica ágil, para estimar el esfuerzo y la duración de tareas.

El protocolo de estimación se lleva a cabo en la reunión de refinamiento, que es uno de los eventos de Scrum que veremos en la próxima lección.

Este juego de planificación se ideó para evitar discusiones dilatadas que no terminan de dar conclusiones concretas.

El funcionamiento es muy simple:

Cada participante dispone de un juego de cartas.

En la estimación de cada tarea, todos muestran el esfuerzo que estiman.

Cuando se considera que éste es mayor de x horas ideales (el tamaño máximo considerado por el equipo para una tarea), se levanta la carta “infinito”.

Las tareas que exceden el tamaño máximo deben descomponerse en subtareas de menor tamaño.

Basado en el hecho de que al aumentar el tamaño de las tareas aumenta también el margen de error, surgió la idea de emplear números de la sucesión de Fibonacci para realizar las estimaciones, de forma que:

El juego de cartas está compuesto por números en sucesión de Fibonacci.

La estimación no se realiza levantando varias cartas para componer la cifra exacta, sino poniendo boca arriba la carta con la cifra más aproximada a la estimación.

Así, si por ejemplo una persona cree que el tamaño adecuado de una tarea es 6, se ve obligado a reconsiderar y, levantar la carta de 5, o bien aceptar una estimación más conservadora y levantar el 8.

En particular, los números de este Planning Póquer son: 0, 1, 2, 3, 5, 8, 13, 21, infinito.

Es frecuente emplear una carta con un símbolo de duda o interrogación para indicar que, por las razones que sean, no se puede precisar una estimación. También es posible incluir otra carta con alguna imagen alusiva, para indicar que se necesita un descanso.

La operativa es la siguiente:

- | Cada participante de la reunión tiene un juego de cartas.
- | Para cada historia de usuario, el producto owner expone la descripción empleando un tiempo máximo.
- | Hay establecido otro tiempo para que el producto owner atienda a las posibles preguntas del equipo.
- | Cada participante selecciona la carta que representan su estimación, y la separa del resto, boca abajo.
- | Cuando todos han hecho su selección, se muestran boca arriba.
- | Si la estimación resulta “infinito”, por sobrepasar el límite máximo establecido, la tarea debe dividirse en subtareas de menor tamaño.
- | Si las estimaciones resultan muy dispares, quien asume la responsabilidad de gestionar la reunión (normalmente suele ser el Scrum Master), puede optar por:
 1. Preguntar a las personas de las estimaciones extremas: ¿Por qué crees que es necesario tanto tiempo?, y ¿por qué crees que es necesario tan poco tiempo? Tras escuchar las razones, repetir la estimación.
 2. Dejar a un lado la estimación de esa tarea y retomar al final o en otro momento aquellas que hayan quedado pendientes.
 3. Pedir al producto owner que descomponga la funcionalidad y valorar cada una de las funcionalidades resultantes.
 4. Tomar la estimación menor, mayor, o la media.

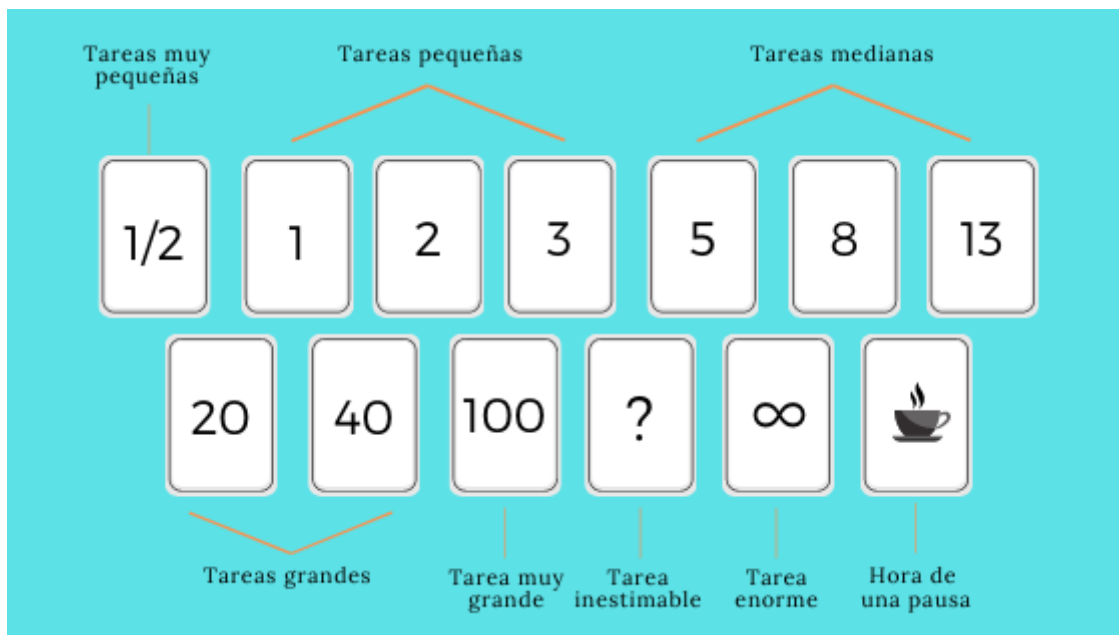


Ilustración 5.1.. ESTIMACION CON PLANNING POKER

Este protocolo de moderación, evita en la reunión los atascos, hace participar a todos los asistentes, consigue alcanzar consensos sin discusiones, y además resulta divertido y dinamiza la reunión de refinamiento, que es donde se lleva a cabo este procedimiento de estimación.

Los beneficios que esta técnica ofrece a los equipos que desarrollan producto son:

- | Promover una mayor cohesión y colaboración en los equipos.
- | Conseguir hacer estimaciones más precisas, basadas en opiniones de diferentes expertos y en el proceso de socializar las estimaciones personales.
- | Detectar más fácilmente la falta de certeza técnica en cuanto al producto, ante la falta de consenso durante la sesión de estimación.
- | Mejorar la comprensión del backlog product.
- | Optimizar el proceso de determinación del valor que cada ítem aporta al negocio frente al coste estimado que tiene su desarrollo.

6. PUNTOS CLAVE

- | Descripción y fundamentos de la medición ágil.
- | Puntos de historia como unidad relativa de medición ágil.
- | Técnicas de estimación ágil.
- | Planning poker como técnica de estimación ágil más extendida.

